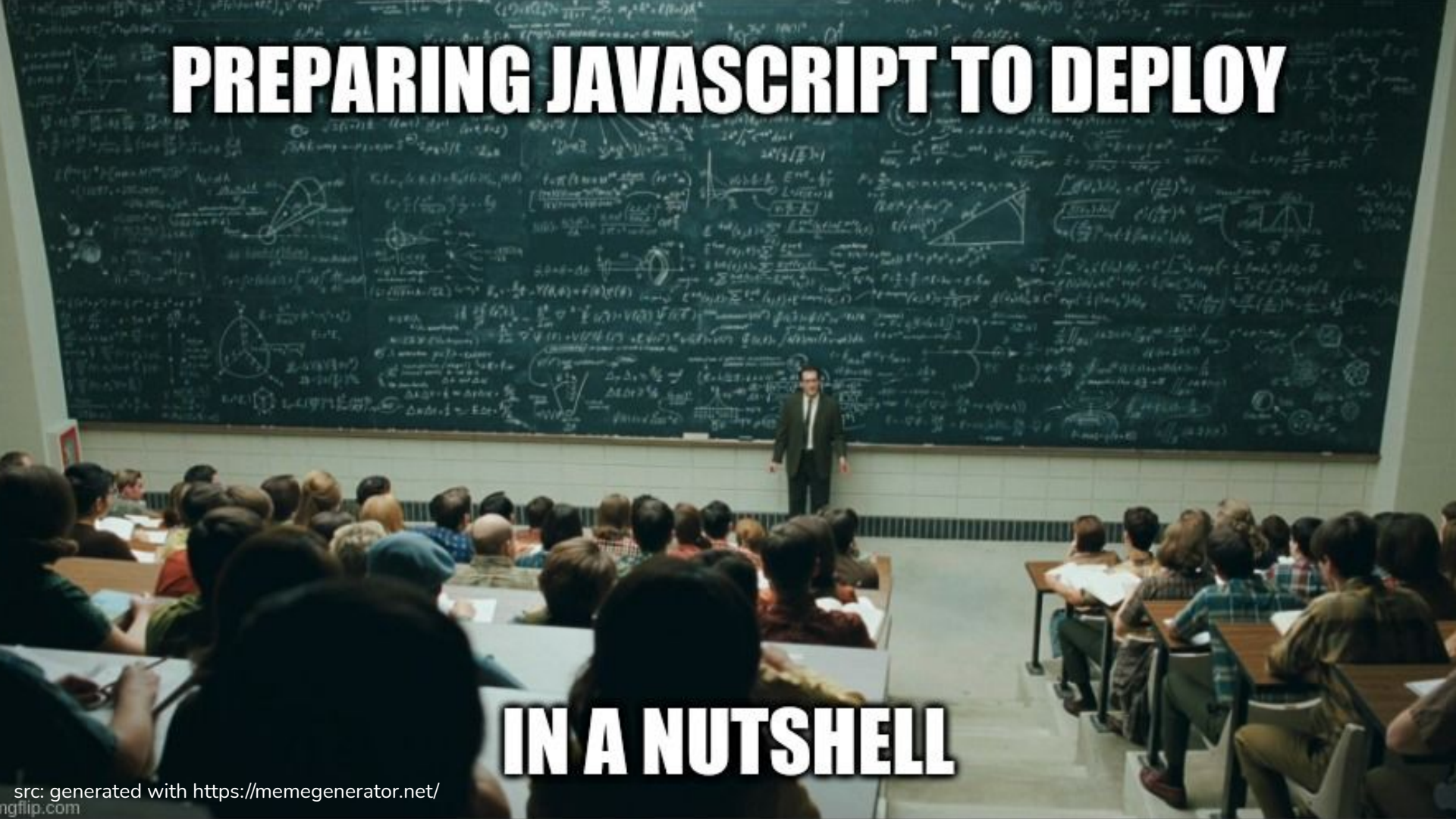
The future is today

Kamil Gałek
BigPicture

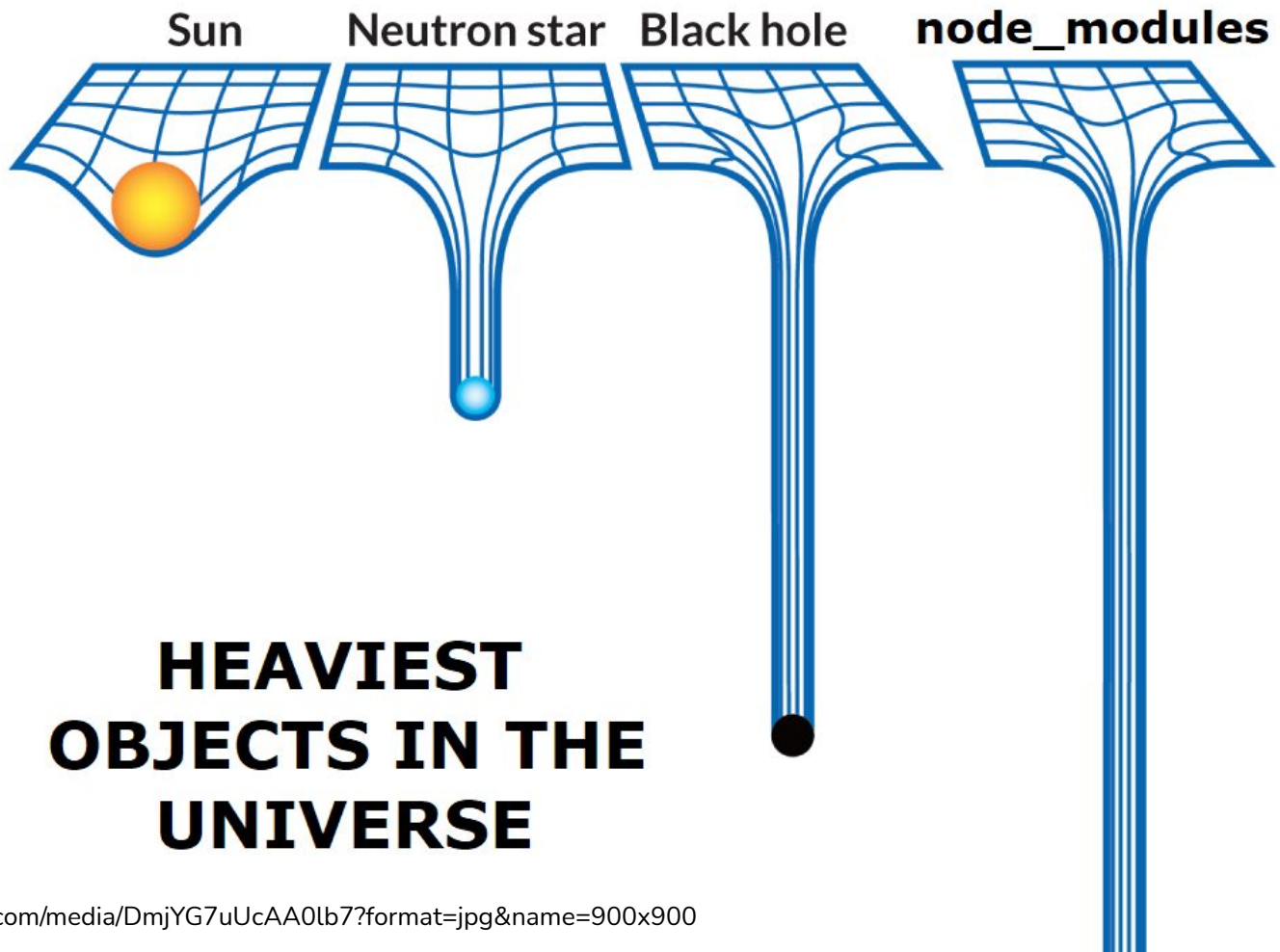
A long, long, time ago…

JS was used to detect
Internet Explorer browser

src: https://miro.medium.com/max/1280/1*8jE0o_rd1ahG2IRADAvgAg.jpeg



src: generated with https://memegenerator.net/



src: https://pbs.twimg.com/media/DmjYG7uUcAA0lb7?format=jpg&name=900x900

src: https://i.redd.it/3bah7kcyhgf71.png

Tomorrow 🙀

Kamil Gałek

Frontend Developer
BigPicture

🚀 The future is today 🚀

esbuild revolution is near

Rules

Questions After - 45 min - Repo

github.com/galczo5/future-is-now

Bundlers?

Why?

Complexity

Browsers

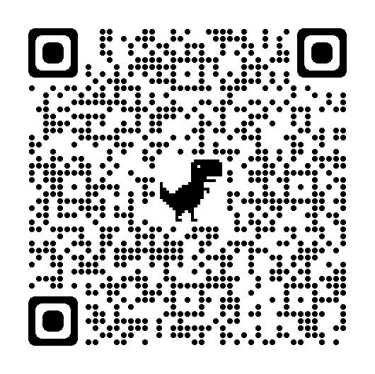
Minify/Uglify?

Size is the most important optimization

Who want to have shorter builds?

Let’s count React developers!

esbuild is production-ready



@angular-devkit/build-angular
use esbuild as a CSS optimizer for
component styles

Angular

Snowpack
The faster frontend build tool.

Alibaba, Intel, Internet Archive

Vite
Next Generation Frontend Tooling

Vue.js

“Talk is cheap. Show me the code.”

Demo #1

node generator - package.json - generation results - dist

Unrealistic examples again 😥

Small apps are not the problem, but nobody develop small apps on production

< 5s

Let’s go for real-world cases!

Demo #2

react generator - small example - huge app

src: https://memegenerator.net/img/images/72255169.jpg

Let’s compare the results

Demo #3

comparator script - comparator demo - node app - react app

Official benchmarks

evanw/esbuild: An extrem

github.com/evanw/esbuild

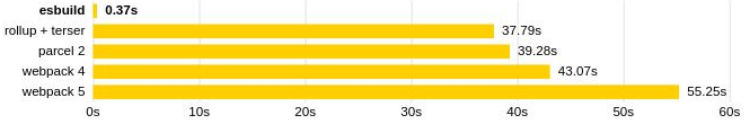
esbuild

An extremely fast JavaScript bundler

Website | Getting started | Documentation | Plugins | FAQ

Why?

Our current build tools for the web are 10-100x slower than they could be:



The main goal of the esbuild bundler project is to bring about a new era of build tool performance, and create an easy-to-use modern bundler along the way.

Major features:

- Extreme speed without needing a cache
- ES6 and CommonJS modules
- Tree shaking of ES6 modules
- An API for JavaScript and Go
- TypeScript and JSX syntax
- Source maps

evanw/esbuild: An extrem

github.com/evanw/esbuild

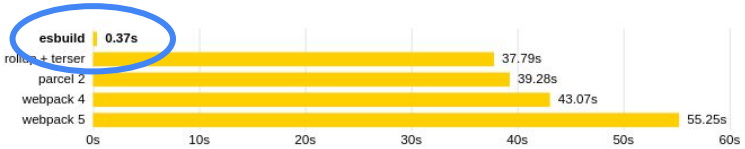
esbuild

An extremely fast JavaScript bundler

Website | Getting started | Documentation | Plugins | FAQ

Why?

Our current build tools for the web are 10-100x slower than they could be:



The main goal of the esbuild bundler project is to bring about a new era of build tool performance, and create an easy-to-use modern bundler along the way.

Major features:

- Extreme speed without needing a cache
- ES6 and CommonJS modules
- Tree shaking of ES6 modules
- An API for JavaScript and Go
- TypeScript and JSX syntax
- Source maps

evanw/esbuild: An extrem

github.com/evanw/esbuild

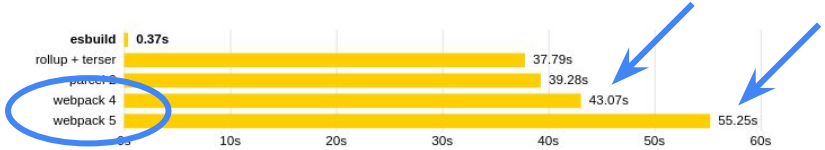
esbuild

An extremely fast JavaScript bundler

Website | Getting started | Documentation | Plugins | FAQ

Why?

Our current build tools for the web are 10-100x slower than they could be:



The main goal of the esbuild bundler project is to bring about a new era of build tool performance, and create an easy-to-use modern bundler along the way.

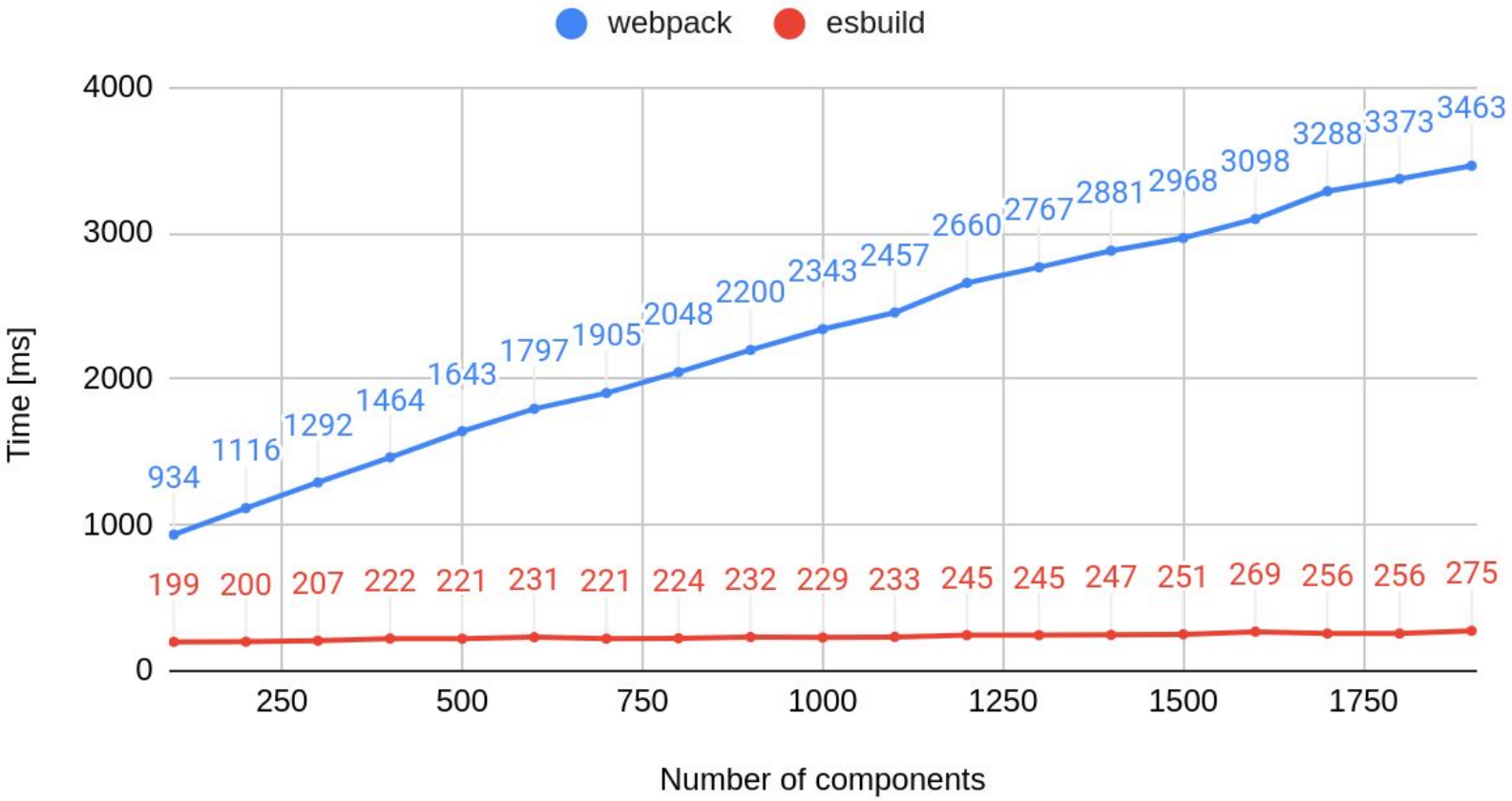
Major features:

- Extreme speed without needing a cache
- ES6 and CommonJS modules
- Tree shaking of ES6 modules
- An API for JavaScript and Go
- TypeScript and JSX syntax
- Source maps

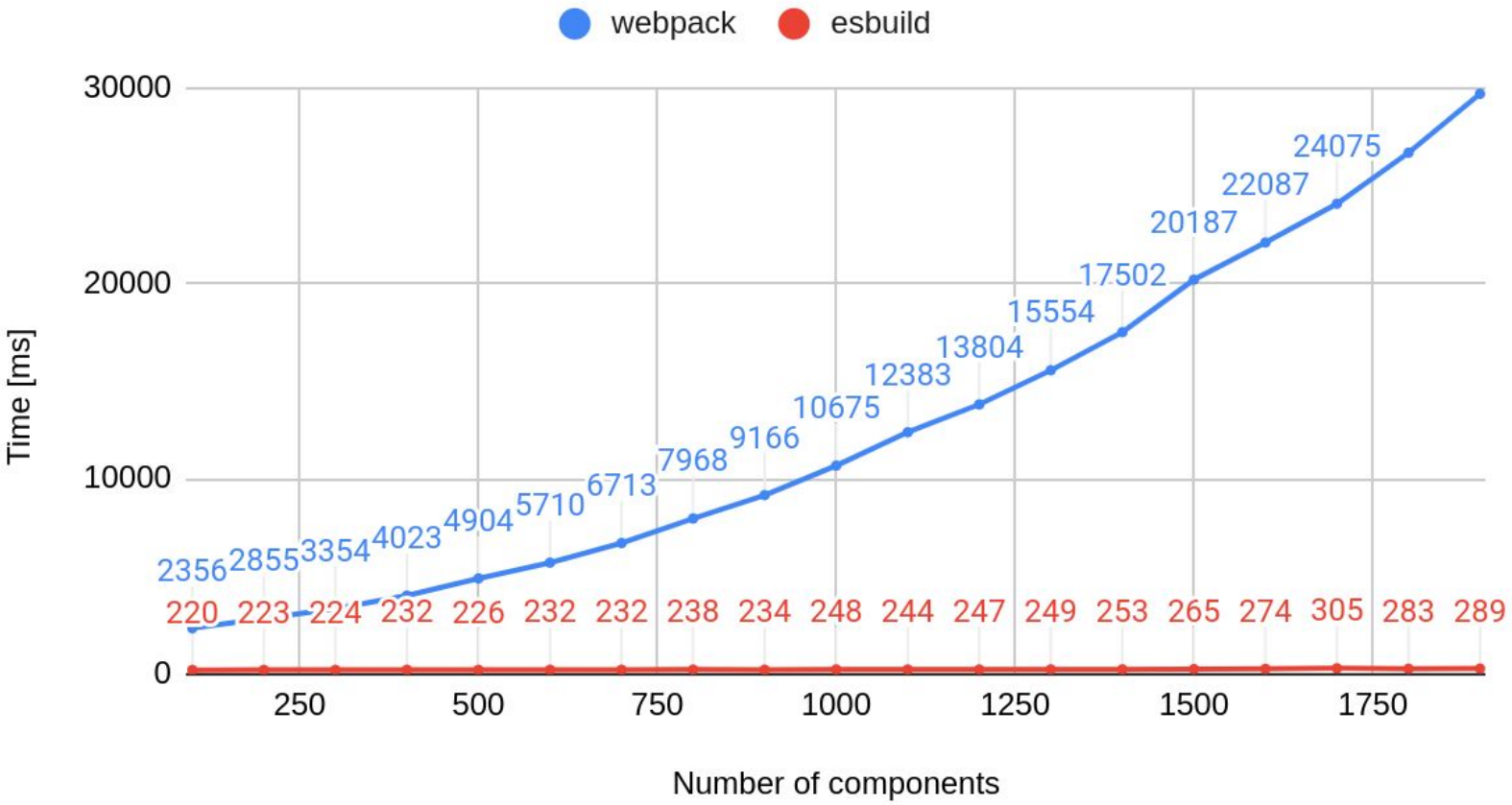
My results

AMD Ryzen 3600 and 32Gb RAM

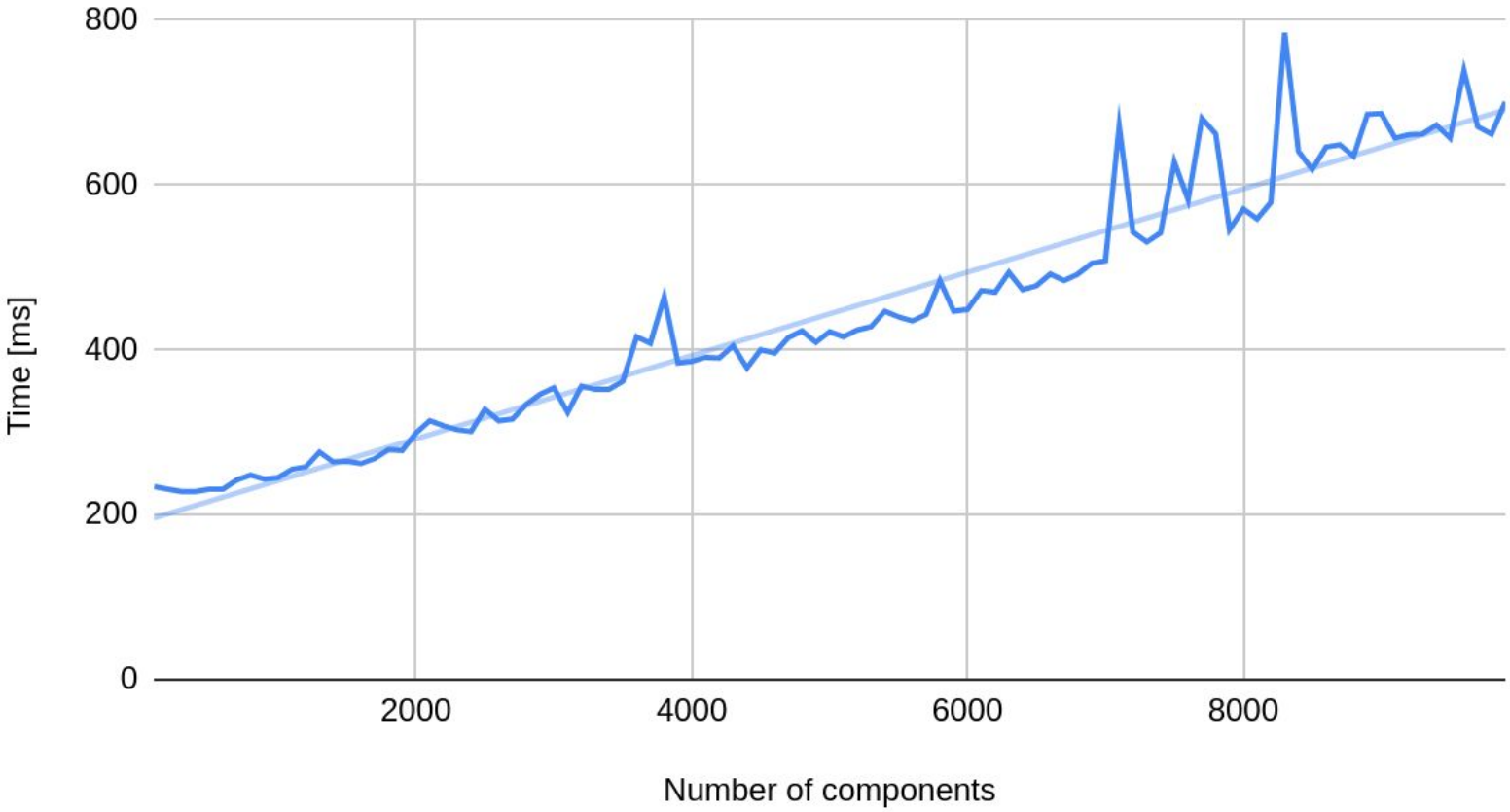
Node app build time



React app build time



React app build time



Let’s be crazy one more time…

src: generated with https://memegenerator.net/

fish /home/kamil/Dev/future-is-now

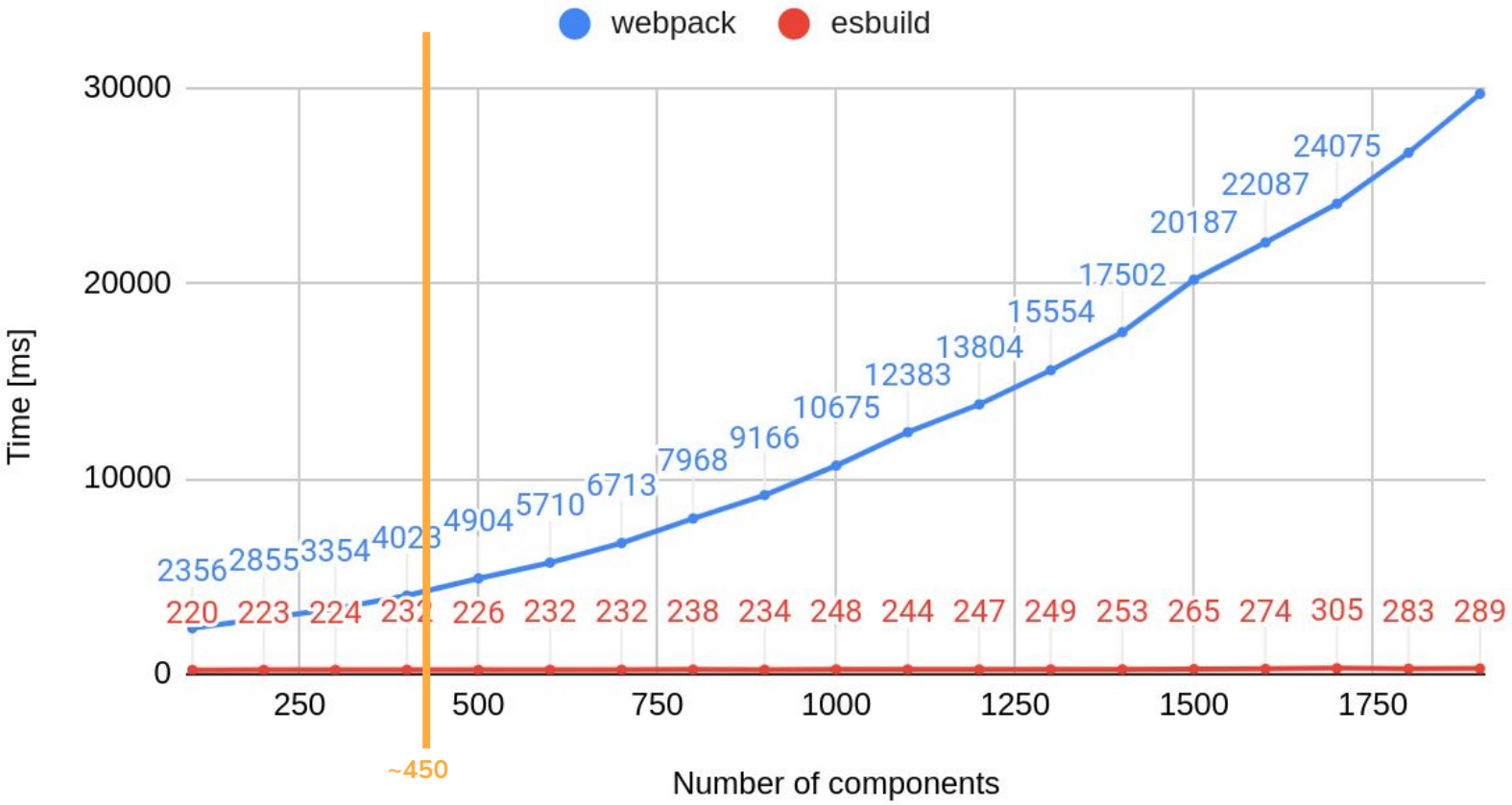
```
generated ./src/react/module99975.jsx
generated ./src/react/module99976.jsx
generated ./src/react/module99977.jsx
generated ./src/react/module99978.jsx
generated ./src/react/module99979.jsx
generated ./src/react/module99980.jsx
generated ./src/react/module99981.jsx
generated ./src/react/module99982.jsx
generated ./src/react/module99983.jsx
generated ./src/react/module99984.jsx
generated ./src/react/module99985.jsx
generated ./src/react/module99986.jsx
generated ./src/react/module99987.jsx
generated ./src/react/module99988.jsx
generated ./src/react/module99989.jsx
generated ./src/react/module99990.jsx
generated ./src/react/module99991.jsx
generated ./src/react/module99992.jsx
generated ./src/react/module99993.jsx
generated ./src/react/module99994.jsx
generated ./src/react/module99995.jsx
generated ./src/react/module99996.jsx
generated ./src/react/module99997.jsx
generated ./src/react/module99998.jsx
generated ./src/react/module99999.jsx
saved ./src/react/main.jsx
saved ./src/react/App.jsx
> ~/D/future-is-now npm run react:esbuild:build

> future-is-now@1.0.0 react:esbuild:build /home/kamil/Dev/future-is-now
> node ./config/react.esbuild.js

esbuild compiled 4198 ms
> ~/D/future-is-now
```

4198 ms

React app build time



Is it possible to run so huge app...?

esbuild

127.0.0.1:4200/index.esbuild.html

ESBUILD

Console

2 1

top

Filter

Default levels

1 Issue: 1

Uncaught SyntaxError: Too many arguments in function call (only 65535 allowed) main.js:9

Failed to load resource: the server responded with a status of 404 (Not Found) favicon.ico:1

100k components

esbuild

127.0.0.1:4200/index.esbuild.html

ESBUILD

Console

2 1

top

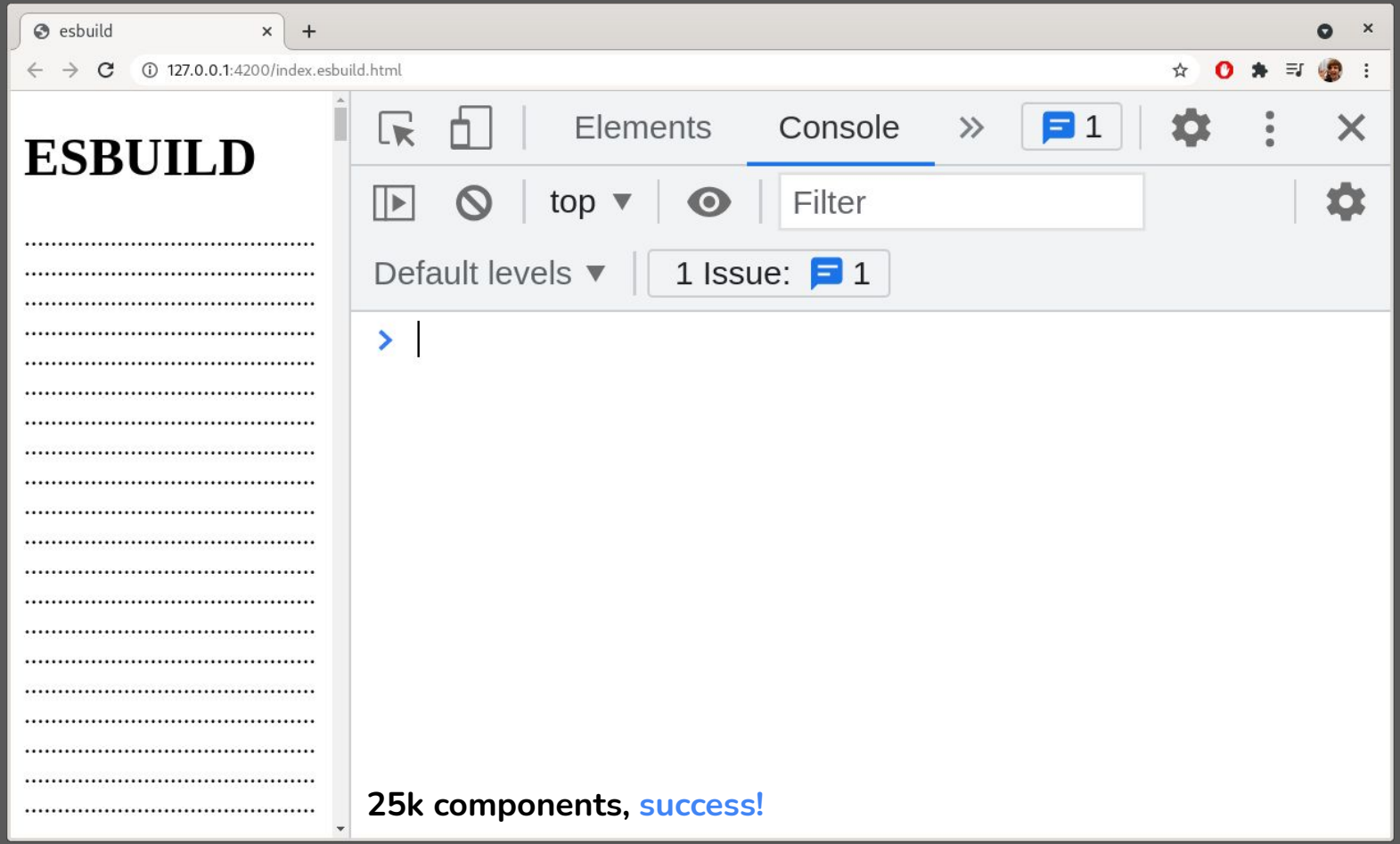
Filter

Default levels

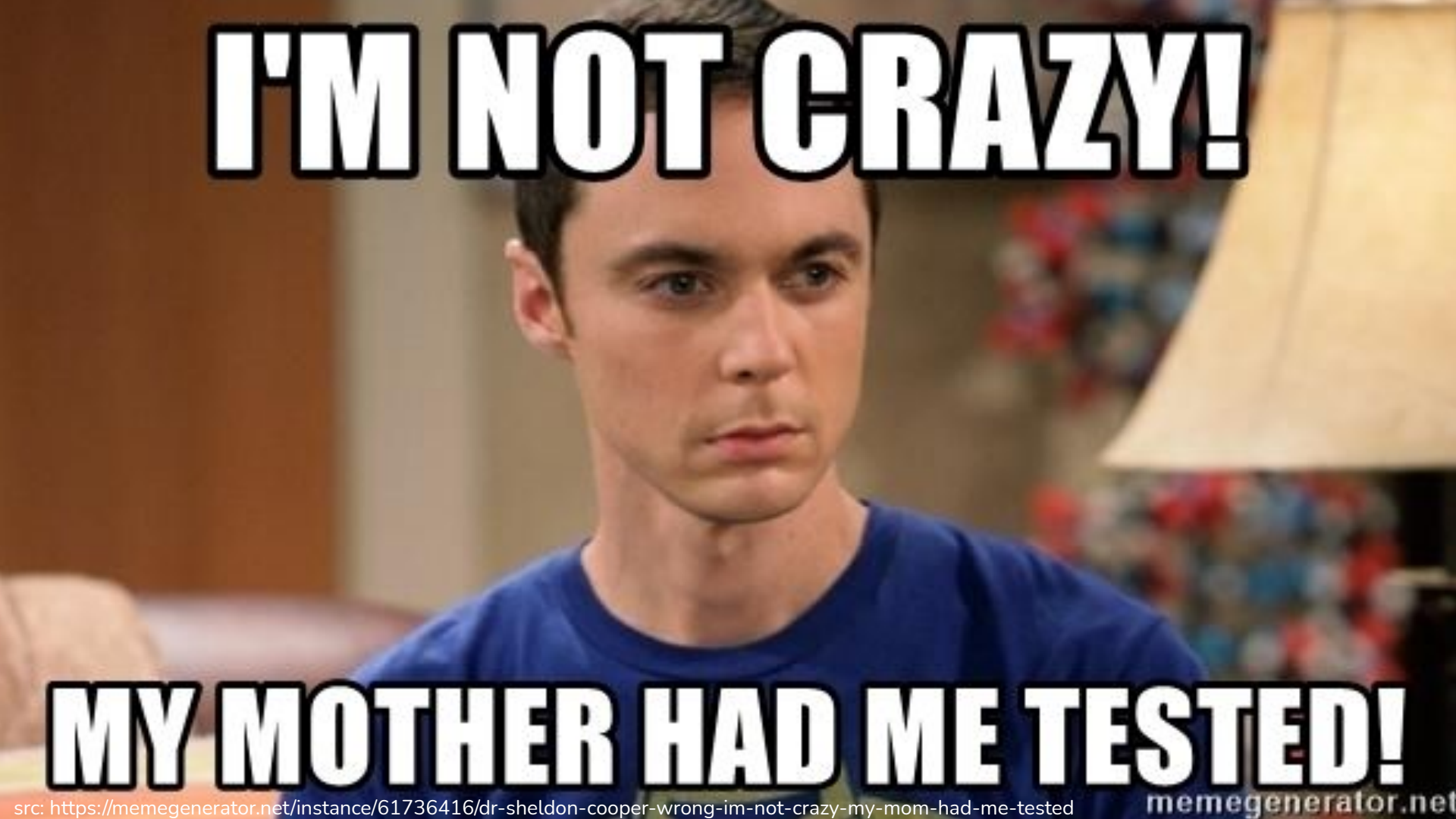
1 Issue: 1

```
    at main.js:9
Uncaught RangeError: Maximum call stack size exceeded    main.js:9
    at nTz (main.js:9)
    at Km (main.js:5)
    at Uf (main.js:9)
    at jf (main.js:9)
    at Hsv (main.js:9)
    at Yt (main.js:9)
    at ve (main.js:7)
    at Jo (main.js:7)
    at sp (main.js:9)
    at main.js:9
```

50k components



25k components, success!

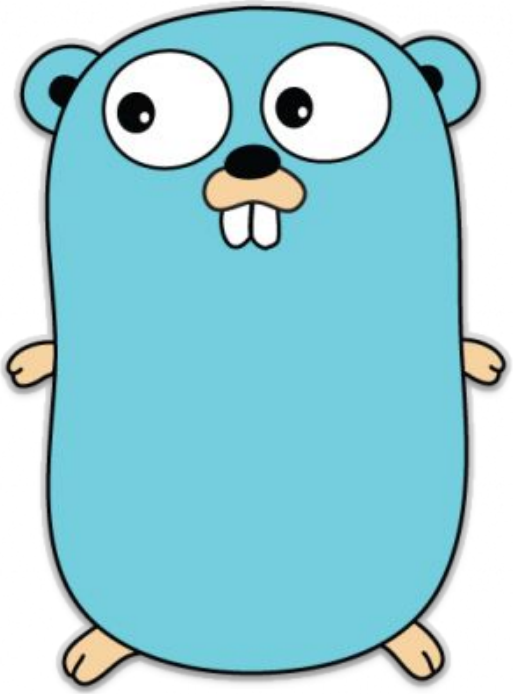


src: https://memegenerator.net/instance/61736416/dr-sheldon-cooper-wrong-im-not-crazy-my-mom-had-me-tested

25k components
1,16s esbuild vs NaN* Webpack

*I resigned after 10 minutes, so we'll never find out how long it takes

How it’s possible?



esbuild - FAQ

esbuild.github.io/faq/#why-is-esbuild-fast

esbuild

Why is esbuild fast?

Several reasons:

- It's written in Go and compiles to native code.

  Most other bundlers are written in JavaScript, but a command-line application is a worst-case performance situation for a JIT-compiled language. Every time you run your bundler, the JavaScript VM is seeing your bundler's code for the first time without any optimization hints. While esbuild is busy parsing your JavaScript, node is busy parsing your bundler's JavaScript. By the time node has finished parsing your bundler's code, esbuild might have already exited and your bundler hasn't even started bundling yet.

  In addition, Go is designed from the core for parallelism while JavaScript is not. Go has shared memory between threads while JavaScript has to serialize data between threads. Both Go and JavaScript have parallel garbage collectors, but Go's heap is shared between all threads while JavaScript has a separate heap per JavaScript thread. This seems to cut the amount of parallelism that's possible with JavaScript worker threads in half according to my testing, presumably since half of your CPU cores are busy collecting garbage for the other half.

- Parallelism is used heavily.

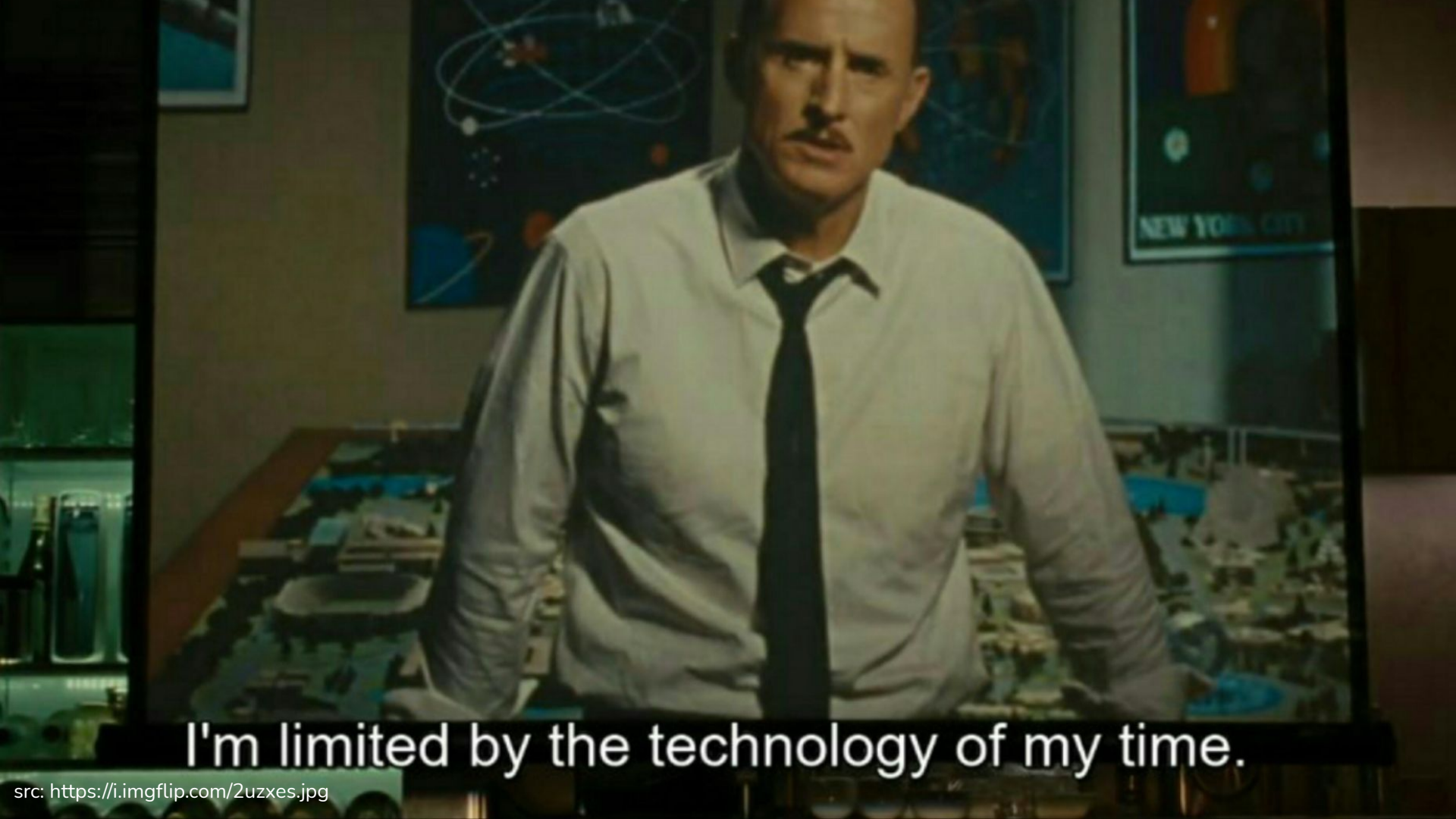
  The algorithms inside esbuild are carefully designed to fully saturate all available CPU cores when possible. There are roughly three phases: parsing, linking, and code generation. Parsing and code generation are most of the work and are fully parallelizable (linking is an inherently serial task for the most part). Since all threads share memory, work can easily be shared when bundling different entry points that import the same JavaScript libraries. Most modern computers have many cores so parallelism is a big win.

- Everything in esbuild is written from scratch.

  There are a lot of performance benefits with writing everything yourself instead of using 3rd-party libraries. You can have performance in mind from the beginning, you can make sure everything uses consistent data structures to avoid expensive conversions, and you can make wide architectural changes whenever necessary. The drawback is of course that it's a lot of work.

  For example, many bundlers use the official TypeScript compiler as a parser. But it was built to serve the goals of the TypeScript compiler team and they do not have performance as a top priority. Their code makes

Why now? Why not earlier?



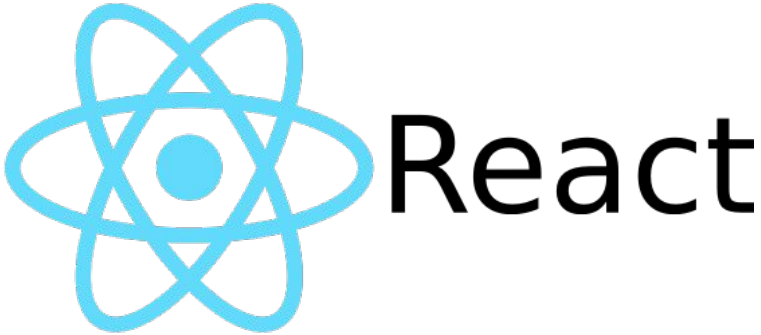
src: https://i.imgflip.com/2uzxes.jpg

Frontend 💖 Javascript

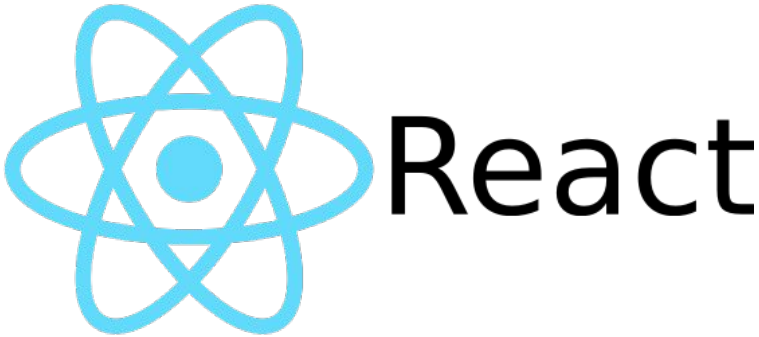
It’s not a bad thing

Where I can use it?

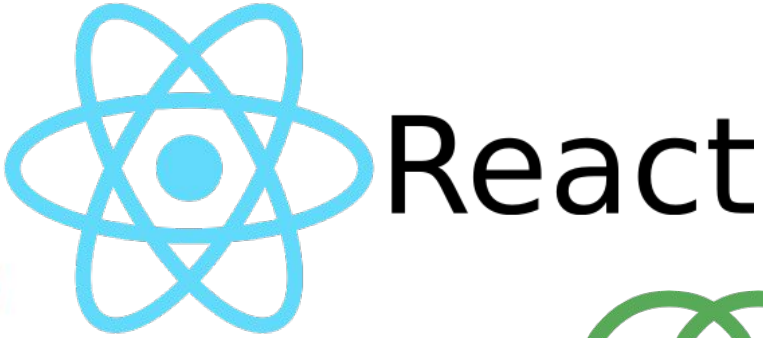

node
JS

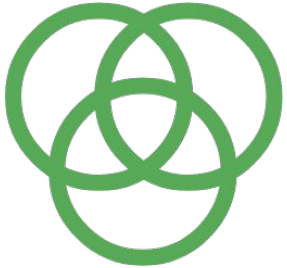

React


node
JS


React


SVELTE





SVELTE

Angular case 😭

Use esbuild for ts/js bundl

github.com/angular/angular-cli/issues/18395

Closed

Use esbuild for ts/js bundling #18395

sp90 opened this issue on 29 Jul 2020 · 4 comments

Maybe esbuild could replace terser & babel for the production builds. But that would only attack a fraction of the actual build time. So in the grand scheme not a order magnitude improvement sadly.

alan-agius4 commented on 29 Jul 2020 Collaborator

Hi **@sod** and **@sp90**,

Thanks for the request and interest in this.

We are following closely the development of ESBuild, Rollup, Webpack and other bundlers.

That being said, we are not planning to move away from Webpack yet. At least not in the near future. In fact currently we actively working on adding support for the next major version of Webpack.

Let's come back to this in the future.

Thanks again 😊

alan-agius4 closed this on 29 Jul 2020

angular-automatic-lock-bot bot commented on 29 Aug 2020

This issue has been automatically locked due to inactivity.
Please file a new issue if you are encountering a similar or related problem.

Read more about our automatic conversation locking policy.

This action has been performed automatically by a bot.

How to build minimal ang

dev.to/marcinwosinek/how-to-build-minimal-angular-12-app-with-esbuild-54cc

DEV

Search...

Log in

Create account

2

2

0

How to build minimal angular 12 app with esbuild

#angular #esbuild #javascript

Marcin Wosinek 9 lip • 2 min read

esbuild (7 Part Series)

1. **How to build minimal angular 12 app with esbuild**
2. How to build angular cli application with esbuild

... 3 more parts...

6. How to lazy load with esbuild

Marcin Wosinek

Javascript developer

Follow

JOINED

3 maj 2021

More from Marcin Wosinek

How to start building a project with esbuild

#javascript #bundling

How to make SolidJS application work from a subfolder

#solidjs #javascript

Day.js vs date-fns

Is it hard to configure?

Demo #4

configuration files

Is it hard to configure?

Not really, if you know what you doing is quite simple.

Roadmap

esbuild - FAQ

esbuild.github.io/faq/#upcoming-roadmap

Upcoming roadmap

These features are already in progress and are first priority:

- Code splitting (#16, docs)
- CSS content type (#20, docs)
- Plugin API (#111)

These are potential future features but may not happen or may happen to a more limited extent:

- HTML content type (#31)
- Lowering to ES5 (#297)
- Bundling top-level await (#253)

After that point, I will consider esbuild to be relatively complete. I'm planning for esbuild to reach a mostly stable state and then stop accumulating more features. This will involve saying "no" to requests for adding major features to esbuild itself. I don't think esbuild should become an all-in-one solution for all frontend needs. In particular, I want to avoid the pain and problems of the "webpack config" model where the underlying tool is too flexible and usability suffers.

For example, I am *not* planning to include these features in esbuild's core itself:

- Support for other frontend languages (e.g. Elm, Svelte, Vue, Angular)
- TypeScript type checking (just run `tsc` separately)
- An API for custom AST manipulation
- Hot-module reloading
- Module federation

I hope that the extensibility points I'm adding to esbuild (plugins and the API) will make esbuild useful to include as part of more customized build workflows, but I'm not intending or expecting these extensibility points to cover all use cases. If you have very custom requirements then you should be using other tools. I also hope esbuild inspires other build tools to dramatically improve performance by overhauling their implementations so that everyone can benefit, not just those that use esbuild.

I am planning to continue to maintain everything in esbuild's existing scope even after esbuild reaches stability. This

Where I can get help?

One more thing…

Other projects worth to follow

Snowpack

Snowpack

So, why I think that **the future** is now?

jQuery revolution

MVC boom

Components

Future > Webpack

(deeper) Future > Future > Webpack

“Frontend is changing every two weeks”

Predictions:

It’s going to be fantastic!

Thank you! 😎